Sichel model, R code

Bayesian Estimation of Earth's Undiscovered Mineralogical Diversity using Noninformative Priors
Journal of Mathematical Geosciences
Grethe Hystad, Ahmed Eleish, Robert M. Hazen, Shaunna M. Morrison, and Robert T. Downs
Corresponding author: Grethe Hystad, Purdue University Northwest, ghystad@pnw.edu

```
### Estimate the population size S and the nuisance parameters
### using the Metropolis algorithm for the Sichel distribution

set.seed(99)

library(rstan) #monitoring convergence
library(gsl)   #gsl wrapper for special functions

z=read.delim("TotMinFeb2014.txt")
#Preparing data
attach(z)
count=sort(count,decreasing = TRUE)
N=sum(count)      #Sample size
w=length(count)   #number of distinct minerals
l=unique(count)   #localitites
j=rev(l)          #localities ordered
locations=length(l)
nj=numeric(locations) #number of minerals with j localities
for (i in 1:locations){
        nj[i]=length(count[count==j[i]])
}

pen = 1.0e20

# adjusting parameters, smaller values give larger acceptance rate
# stdev for Metropolis algorithm for S
st.S = 50

# stdev for Metropolis algorithm for a
st.a = 0.01

# stdev for Metropolis algorithm for b
st.b = 0.1

# stdev for Metropolis algorithm for gamma
st.gamma = 0.01
```

```
#to monitor accpetance rate
ac=0

#Negative loglikelihood function
negLogLikelihood = function(a, b, gamma, S) {
if(a <= 0.0 | b <= 0.0){
        return(pen)
}

LogFacN=numeric(locations)
for(i in 1: locations){
if(j[i]<50) {
    LogFacN[i] = log(fact(j[i]))
    } else {
    LogFacN[i] = lngamma(j[i]+1)
        }
}
omega = sqrt(b*b + a*a) - b
B1=bessel_lnKnu(abs(gamma+j), a)
B2=bessel_lnKnu(abs(gamma), omega)
temp1=j*log((omega*b)/a)
temp2=gamma*log(omega/a)
logj=B1-B2+temp1+temp2-LogFacN
LogP = sum(nj*logj)
LogPP = LogP - sum(lnfact(nj))
Log0=bessel_lnKnu(abs(gamma),a)-B2+temp2
Log1 = (S - w)*Log0
Log2 = - lnfact(S - w)
Log3 = lnfact(S)
LogL = LogPP + Log1 + Log2 + Log3
return(-LogL)
}

#Metropolis algorithm
#Proposal distribution
metropolis=function(a,b,gamma,S,st.a, st.b, st.gamma, st.S){
a_star=abs(rnorm(1,a,st.a))
b_star=abs(rnorm(1,b,st.b))
gamma_star=rnorm(1,gamma,st.gamma)
repeat{
S_star=floor(abs(rnorm(1,S,st.S)))
if(S_star>w-1)
break
}
logr=negLogLikelihood(a,b,gamma,S)-negLogLikelihood(a_star,b_star,gamma_star,S_star)
```

```
r=exp(logr)
# accept or reject proposed value
if (runif(1)<min(r,1)){a=a_star; b=b_star; gamma=gamma_star; S=S_star; ac=ac+1
} else{
        a=a; b=b; gamma=gamma; S=S; ac=ac
        }
return(c(a=a, b=b, gamma=gamma, S=S, ac=ac))
}

#Number of chains
chains = 3
#Number of iterations
iter = 1000000

simulations = array(NA, c(iter, chains, 4))
dimnames(simulations) = list(NULL, NULL, c("a","b","gamma", "S"))
for (kk in 1:chains){
        # starting values
        a=0.08+(kk-1)*2*st.a
        b=0.8+(kk-1)*2*st.b
        gamma=-0.52+(kk-1)*2*st.gamma
        S=6118+(kk-1)*2*st.S
        for (t in 1:iter){
                temp = unlist(metropolis(a,b,gamma,S,st.a, st.b, st.gamma, st.S))
                a = as.numeric(temp[1])
                b =  as.numeric(temp[2])
                gamma = as.numeric(temp[3])
                S = as.numeric(temp[4])
                ac = as.numeric(temp[5])
                simulations[t,kk,] <- c(a,b,gamma,S)
        }

}
monitor(simulations,digit=4)
```